Measures of Similarity in Memory-Based Collaborative Filtering Recommender System – A Comparison

Shalini Christabel Stephen
Murdoch University
School of Engineering and Information Technology
Murdoch University, South Street Campus, WA
shalz72002@yahoo.co.in

Dr. Hong Xie
Murdoch University
School of Engineering and Information Technology
Murdoch University, South Street Campus, WA
+6193606087
H.Xie@murdoch.edu.au

Shri Rai
Murdoch University
School of Engineering and Information Technology
Murdoch University, South Street Campus, WA
+6193606087
s.rai@murdoch.edu.au

ABSTRACT

Collaborative filtering (CF) technique in recommender systems (RS) is a well-known and popular technique that exploits relationships between users or items to make product recommendations to an active user. The effectiveness of existing memory based algorithms depend on the similarity measure that is used to identify nearest neighbours. However, similarity measures utilize only the ratings of co-rated items while computing the similarity between a pair of users or items. In most of the e-commerce applications, the rating matrix is too sparse since even active users of an online system tend to rate only a few items of the entire set of items. Therefore, co-rated items among users are even sparser. Moreover, the ratings a user gives an individual item tells us nothing about his comprehensive interest without which the generated recommendations may not be satisfactory to a user. In order to be able to address these issues, a comprehensive study is made of the various existing measures of similarity in a collaborative filtering recommender system (CFRS) and a hierarchical categorization of products has been proposed to understand the interest of a user in a wider scope so as to provide better recommendations as well as to alleviate data sparsity.

CCS Concepts

Human-centered computing~Empirical studies in collaborative and social computing • *Applied computing~Online shopping*

Keywords

Recommender systems; collaborative filtering; Pearson correlation coefficient; adjusted cosine similarity; slope one predictor; hierarchical categorization.

MISNC '17, July 17-19, 2017, Bangkok, Thailand © 2017 Association for Computing Machinery.ACM ISBN 978-1-4503-48812/17/07…$15.00
DOI: http://dx.doi.org/10.1145/3092090.3092105

1. INTRODUCTION

The use of the Internet started spreading rapidly in the mid 1990's and from thereon there has been an explosive growth of e-commerce and online environments in the past two decades [1]. A huge number of companies market and sell their products through the Internet [2, 3]. Today e-commerce has paved way for companies to present consumers with more choices on products and has grown into a multi-billion dollar industry. However, its downside is the overwhelming amount of information that consumers are exposed to making it difficult for them to select products that best fit their needs [4]. This scenario has given rise to a class of web applications called recommender systems. These systems provide automated and personalized suggestions of products to customers and assist them to make buying decisions [4, 5]. Recommender systems are algorithms that make use of a particular user's preference patterns and turn them into predictions of that user's possible future likes and interests that are then suggested to the user [6]. Recommendations made by such systems can help users navigate through large information space of product descriptions, news articles or other items [7, 8]. The huge potential of recommender systems was first noticed by web entrepreneurs, however, it is not limited to e-commerce alone, but is also a topic of burgeoning interest in the fields of mathematics, physics and psychology [6]. The objective of a recommender system is this: to alleviate information overload by using some personalization techniques adapted for a specific user. Subsequently the user is then presented with the most relevant and attractive product.

Broadly, there are two ways that patterns of user interest in products can be analyzed and recommendations generated: 1) Content based filtering: this is an important class of recommender systems and the most obvious approach used to tackle the problem of information filtering. Product recommendations are made based on correlations between the description of the items' and users' preferences. However, content-based filtering has some limitations; while it is relatively easier to use the content based approach for recommending items containing textual information, it would prove to be expensive and time consuming to be used for recommending items like video, music tracks etc., since the required domain knowledge may not be readily available or straightforward to maintain [9]. Content-based filtering techniques cannot generate serendipitous finds (different types of items), i.e., the system recommends more of what the user has already seen

before. Moreover, content-based filtering methods cannot filter items based on assessment of quality, style or point of view [10]. 2) Collaborative filtering: this is a complementary filtering technique that is based on the intuition that when recommendations are needed for a particular user, the basic approach would be to select items favoured by other users similar to the target user [6]. While content-based techniques rely on features of the item or user to make predictions and product recommendations, collaborative filtering uses only the user-item rating data to make predictions and recommendations. This is a significant difference between the two techniques [11]. In a real life scenario, while searching for products, people often rely on the suggestions given by friends/people more experienced in the area. Essentially, collaborative filtering (CF) recommender system automates such 'word of mouth' recommendations [12]. A typical collaborative filtering recommender system (CFRS) would not use any information regarding the content of an item (e.g., description, author) but are rather based on the preference patterns of other users. It addresses some of the shortcomings of the content-based filtering technique, and is suitable for a wide variety of online applications like recommending books, movies, restaurants, jokes, grocery products etc. [2, 12, 13].

In reality however, both approaches may be needed to make predictions and recommendations of a better quality [11]. Consequently, hybrid techniques such as "collaboration via content" [2], [14], "content-boosted" collaborative filtering algorithms [15] were developed which are a combination of the collaborative filtering and content-based techniques thereby avoiding the limitations of either approach.

There are two general classes of CF algorithms: *Memory-based* algorithms (also known as nearest neighbour-based algorithms) operate over the entire user database to make predictions. They are one of the most successful approaches to recommendations and have been studied extensively and found various applications in e-commerce [6]. However, they have several limitations. For example, the similarity values used in this technique are based on common items and therefore the recommendations are unreliable when data are sparse and common items are few [11]. *Model-based* algorithms use the user database to train the system in order to ultimately create a user model which is then used for predictions [12]. Generally, since the model-based algorithms have the ability to describe the various aspects of the data, they tend to provide more accurate results than memory-based algorithms. However, most commercial systems such as Amazon and TiVo use the memory-based algorithms due to their relative simplicity and their abilities to provide intuitive explanations of the reasoning behind the recommendations, which enhance the user experience beyond improved accuracy. In addition, the memory-based algorithms are capable of providing immediate recommendations using newly entered user feedback [9] while avoiding the complications of potentially expensive model building stage. Nevertheless, intuitive explanations and immediate recommendations alone are not enough to ensure the success of a recommender system, accuracy of the recommendations is also of importance. This is because even though a customer may understand the reasoning behind the recommendations, he would soon lose interest if the recommendations are inaccurate.

Generally when people shop online, they react to products, and the aim of the recommender system is to mine this hidden resource for information [16]. User reaction can be captured in one of the following two methods: *explicit feedback* and *implicit feedback*. Explicit feedback of a user on a product (item) refers to the rating given *explicitly* by a user to a product (item) to show his level of interest in that product or item. Implicit feedback on the other hand is obtained indirectly by capturing the interaction of a user with an item such as purchase history, browsing history or search patterns [12] [9]. Since explicit feedback captures user preferences directly and in a granular way, they are more precise but are difficult to collect from users due to the reluctance of users to rate products and therefore may not always be available. Implicit feedback is easier to collect, but they are less accurate in reflecting user preferences. Regardless of the type of user feedback, item predictions need to be made based on the interest shown to the entire collection of items by similar users [17].

In this study, the various measures of similarity are compared with each other and the following observations have been made:

1) Since the similarity measures used by the CF algorithms make use of co-rated items to compute similarity and since the rating matrix is already sparse, there is not enough information to base a prediction on.
2) While calculating similarity between users or items, only individual items rated by a user are used which tells us nothing about the user's comprehensive interest in products.

We propose to address these challenges by exploring the following technique:

Computing the similarity between two items or users even in the absence of users who have rated both items commonly.

Exploring information such as which type of items the user has chosen to rate and using this information to generate recommendations.

The rest of the paper is organized as follows: Section 2 describes the general working of a memory-based approach and elaborates on the steps involved in the generation of recommendations. Section 3 gives a brief overview of the way the different memory-based algorithms discussed in previous section can be evaluated followed by the results and discussion in section 4 and conclusion in section 5.

2. NEIGHBOURHOOD BASED APPROACH – A FRAMEWORK

The task of a CFRS is to predict the utility of an item to a target user based on the feedback given by other similar users to that item and to provide useful personalized recommendations of items that might interest the target user [4, 17]. In the user-item matrix every user-item pair has a value representing the degree of preference a user has for a particular item, denoted as r_{ui} corresponding to the feedback given by user u to item i. The matrix also has missing values where users have not given their preferences for certain items. The task of a memory-based CFRS is to predict a user preference and fill in the missing values in the user-item matrix. To predict a value for the missing preferences, memory based collaborative algorithms firstly need to compute the similarity weights between every pair of user in the database. The fundamental assumption is that if user *u* and *v* rate *n* items similarly, or have similar behaviour e.g. browsing, buying a similar set of items, then they are likely to act on other items similarly. Secondly,

the predicted rating for a missing value is calculated which is a weighted sum of all the ratings of the users in the database.

The memory-based CF algorithm works in three steps:

1) Calculate the similarity $w(u,v)$ or $w(i,j)$ which reflects distance, correlation or weight between two users or two items respectively.
2) Produce a prediction p_{uj}, for the target user u, by taking a weighted average of all the ratings of the user or the item in case of item similarity method.
3) Generate top N recommendations for the target user.

In the following, these three steps are elaborated and the different methods of similarity calculation between users or items is discussed along with other factors that play a role in the accuracy, efficiency and the overall quality of the memory-based CF algorithm.

2.1 Measures of Similarity

The different types of memory-based algorithms differ in the details of the weight calculation between users or items.

2.1.1 Correlation

The concept of collaborative filtering (CF) to recommend products online was first introduced by the developers of the Tapestry system in 1992 [20]. In their system explicit opinions of people (qualitative annotations), regarding a particular product was used to suggest potential products to target users within a close knit community. [10, 21]. However, while implementing the principle of collaborative filtering in a larger community, it cannot be expected of people to know each other. Therefore, in 1994, a ratings-based CF technique was used by the GroupLens system. Here, the explicit user feedback, r_{ui}, was used to measure the extent to which the interest scores of two users linearly relate with each other. In this context, Resnick et al [16] were the first to utilize the Pearson Correlation Coefficient (PCC) as the basis for the weights of user similarity. It can be defined as:

$$w(u,v) = \frac{\sum_{i \in I_{uv}}(r_{ui}-\bar{r}_u)(r_{vi}-\bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}}(r_{ui}-\bar{r}_u)^2}\sqrt{\sum_{i \in I_{uv}}(r_{vi}-\bar{r}_v)^2}} \quad (1)$$

where I_{uv} is the set of items rated by both user u and v. The mean rating of user u can be defined as:

$$\bar{r}_u = \frac{1}{|I_u|}\sum_{i \in I_u} r_{ui} \quad (2)$$

where I_u is the set of items user u has rated and I_v is the set of items rated by user v.

The accuracy of the correlation computation depends on the size of the set I_{uv}. The more the number of co-rated items, the higher the accuracy. The value of the correlation ranges between -1 and 1 (inclusive) where 1 means total positive correlation, 0 means no correlation and -1 is total negative correlation. A correlation value of 0 cannot be used to make a rating prediction. Generally, negative rating correlations are also not used in rating prediction since they are less reliable than positive ones and do not provide any significant information to the prediction. However, it would be useful to investigate if negatively correlated groups can indicate whether these groups are compatible for other categories of items [9, 22]. Rating prediction is made based on the weighted sum of ratings of the users who have a positive correlation with the target user. However, not all users having a positive correlation are chosen, only k nearest neighbours are selected for the formation of a neighbourhood. The candidate neighbours can be limited in one of two ways as given below:

Top N-filtering: where for each user, only the N-nearest neighbours' ratings are considered. The choice of N is crucial, since choosing a large value will require excessive memory and will also slow down the rate of prediction. On the other hand choosing a small value may reduce the coverage of recommendations.

Threshold filtering: This approach does not keep a fixed number of nearest neighbours like the previous method, but considers all those neighbours whose similarity weight has a magnitude greater than a given threshold.

The Pearson Correlation coefficient has been found to work better than other measures of similarity [23] and is one of the most accurate and popular memory-based scheme [24]. However, most of the real world application datasets have millions of items and very few user ratings overlap with each other. Since PCC takes into account only the set of items that are co-rated by two users, similarity cannot be calculated if there are no intersections of rated item sets between two users.

2.1.2 Vector Similarity

Vector similarity, also known as cosine similarity, is a measure of similarity between two vectors. This metric is used frequently to determine similarity between text documents by treating each document as a vector of word frequencies and computing the cosine of the angle formed by the two frequency vectors [12, 25]. This method has also been adopted in collaborative filtering. Formally, if there is a $U \times I$ user-item matrix, then the similarity between two users u and v is defined as the angle between the interest vector of user u and the interest vector of user v. Here, users take the role of documents, items take the role of words and ratings take the role of word frequencies [12]. It is defined as:

$$w(u,v) = \sum_{i \in I_{uv}} \frac{r_{ui}}{\sqrt{\sum_{k \in I_u} r_{uk}^2}} \frac{r_{vi}}{\sqrt{\sum_{k \in I_v} r_{vk}^2}} \quad (3)$$

where I_{uv} is the set of items rated by both user u and v, I_u is the set of items user u has rated and I_v would be the set of items rated by user v.

When users express their interest in products through explicit feedback, different users may use different rating scales, in other words, users have tendencies towards higher or lower ratings. For instance, consider the following utility matrix where users have given their preferences on a scale of 1 - 5:

Table 1. *Grade inflation in user ratings*

	U4	**U5**	**U6**
It4	3.5	2	-
It5	2	3.5	4.5
It6	-	4	4
It7	4.5	-	5
It8	5	2	5
It9	1.5	3.5	4.5
It10	2.5	-	4
It11	2.	3	4

Here, the user U_5's lowest rating is 2 and the highest is 4. The user seems to be avoiding extremes. On the other hand user U_6's ratings range from 4 to 5. So now, comparing two users becomes a bit tricky. Two users' lowest and highest ratings differ and this variability can create a problem with recommendation system. This is known as grade inflation where different users use different scales. Cosine similarity does not account for such differences in the rating behaviour of users. This problem is handled by using the Pearson correlation coefficient or the adjusted cosine similarity measure [25] since both these measures normalize the ratings by considering the deviation of a user rating from their mean. The two most popular methods used for normalizing the ratings are mean-centering and Z-score. The mean-centering determines whether a rating is positive or negative.

$$m(r_{ui}) = r_{ui} - \bar{r}_u \quad (4)$$

The Z-score shows the relationship of a rating to the mean rating. So whereas the mean removes the offsets caused by different rating behaviour, the Z-score also considers the spread in the individual rating scales.

$$z(r_{ui}) = \frac{r_{ui} - \bar{r}_u}{\sigma_u} \quad (5)$$

σ_u – standard deviation of the ratings of user *u*

Initial investigations had found both mean-centering and Z-score to give comparable results, however a more recent study shows Z-score to work better [4].

2.1.3 Item-based Similarity

User-based correlation, the technique discussed above, faces some challenges. To find the most like-minded user matches, a target user needs to be compared with every other user and hence the number of computation increases with the number of users. If the algorithm could be made to spend less time searching for neighbours, then the recommendations can be generated quickly, but this worsens the quality of recommendations. In order to overcome these issues, Sarwar et al [21] proposed an item-based approach to CF. This approach works on the same principle as user-based CF, but measures similarity scores between items instead of users. This allows offline computation of similarity between items allowing to shorten the time needed to obtain recommendations, i.e. similarity can be computed ahead of time and combined with a user's rating to generate recommendations. Vector similarity is used to measure the similarity scores between items. However, in an effort to account for grade inflation in user ratings, Sarwar in his paper proposed the adjusted cosine similarity that offsets the drawback of the cosine similarity by subtracting the corresponding user average from each co-rated item pair. Formally, the similarity between two items *i* and *j* can be defined as:

$$w(i,j) = \frac{\sum_{u \in U}(r_{ui} - \bar{r}_u)(r_{uj} - \bar{r}_u)}{\sqrt{\sum_{u \in U}(r_{ui} - \bar{r}_u)^2}\sqrt{\sum_{u \in U}(r_{uj} - \bar{r}_u)^2}} \quad (6)$$

The difference between (1) and (8) is, while $\bar{r}_u$ and $\bar{r}_v$ in (3) are the mean ratings of user *u* and *v* over their co-rated items only, $\bar{r}_u$ and $\bar{r}_v$ in equation (6) are taken over all ratings of user *u* and *v*.

Empirical evidence presented by [21] and [26] shows that item-based algorithms can provide better computational performance than user-based collaborative methods. Still, the choice of implementing a user-based or an item-based CF would depend on some of the following important criteria. As mentioned earlier, the *accuracy* of predictions in memory-based systems depend on the size of the set I_{uv}. Generally, size of the set I_{uv} is greater in item-based approach than user-based approach. In other words, the average number of common ratings between any two items is more compared to the average number of common ratings between any two users in a user-based approach making the item-based similarity approach superior in terms of accuracy. The *memory* and *computational efficiency* depend on the ratio between the number of users and items. The time and memory required to compute the similarity weights of item based systems is lesser in applications where the number of users exceeds the number of items. Therefore, if the number of items is going to be static, the item-based CF algorithm may be more efficient, also the similarity weights can be measured at infrequent time intervals. However, the downside to this technique is that, since the predicted rating for an item is based on the ratings given to similar items, the recommendations will mostly contain items that are related to what the user appreciates often. Whereas, the user-based method offers serendipitous recommendations [4].

2.1.4 Slope One Predictor

Slope One Predictor algorithm [24] is a popular algorithm for item-based CF. A major advantage of this algorithm is its simplicity and ease of implementation. In this scheme, given the user rating for one item, the algorithm predicts the rating for a non-rated item for the same user. This is done by exploiting the "popularity differential" between items. It subtracts the average ratings of two items to measure how much more, on average, one item is liked than another by the user population. This information is used to predict a rating for non-rated item. The average deviation of item *i* with respect to item *j* is defined as:

$$dev_{i,j} = \frac{\sum_{u \in U_{ij}} r_{ui} - r_{uj}}{|U_{ij}|} \quad (7)$$

where U_{ij} is the set of users who have rated both items *i* and *j*. The validity of the deviation depends crucially on U_{ij}, the larger the overlap, the more reliable the value.

In the above described methods, the predicted rating for a particular user is based only on a small collection of ratings on common items. Therefore, the recommendations generated may be poor if the variability of user ratings differ substantially among neighbours. Hence, it becomes necessary to account for the *significance of a similarity weight*. This is achieved by reducing the magnitude of a similarity weight by a factor proportional to the number of commonly rated items if this number is less than a given parameter $\gamma > 0$.

$$\bar{w}_{uv} = \frac{\min\{|I_{uv}|\}}{\gamma} \times w_{uv} \quad (8)$$

where I_{uv} is the set of items rated by both user *u* and *v*

w_{uv} as given in (3)

γ is a non-zero constant

Another aspect that is important to be accounted for is *popularity bias*. For example, most people would like the movie 'The Godfather', so basing the weight computation on such movies would produce artificially high values. Similarly, a user who always rates items in the same way provides less predictive information than one whose preferences vary from one item to

another. An approach that is helpful and found to have improved the predictive accuracy is the Inverse User Frequency. A weight γ_i is given to each item *i* in proportion to the log ratio of users that have rated *i*.

$$\gamma_i = \log\frac{|U|}{|U_i|} \quad (9)$$

where U is the set of all users in the dataset and U_i is the set of users who have rated item *i*.

2.1.5 Union-Based Similarity

In all of the previously discussed similarity measures the distance between any two users cannot be calculated in the absence of overlapping items. However, by ignoring single rated items especially in the case of sparse datasets, significant information is discarded [22]. Moreover, two users or items can be similar even if they do not have overlaps. Patra et al [27] propose a new similarity measure that does not depend on co-rated items unlike other measures and it uses the Bhattacharyya measure that has been already used in signal processing, image processing and pattern recognition. The Bhattacharyya measure is used to measure the divergence of two probability distributions. In the context of collaborative filtering, this measure is used to find similarity in the rating pattern of two items (BC coefficient) which is then combined with the local similarity between the ratings on the pair of items. Their proposed measure termed as Bhattacharyya Coefficient in CF (BCF) is defined as:

$$S(u,v) = \sum_{i\in I_u}\sum_{j\in I_v} BC(i,j)loc(r_{ui}, r_{vj}) \quad (10)$$

In this definition, the BC coefficient, $BC(i,j)$ measures similarity between the global rating distribution of two items, giving the similarity in the popularity if the items. For example let I = (0, -, 0.5, -, 0, -, 0.5, -, 1, -) and J = (-, 0, 0.5, - ,0, -, 0.5,-,1) be the rating vectors of items I and J, respectively. The ratings lie in {0, 0.5, 1 and - represents unknown values then the BC coefficient is formulated as:

$$BC(I,J) = \sum_{h=1}^{m}\sqrt{I_h J_h} \quad (11)$$

where *h* represents the bins, *m* is the total number of bins and $I_h = \frac{\#h}{\#i}$, #i is the number of users who rated item *i*, #h is the number of users who rated the item *i* with rating value *h*.

Similarly $J_h = \frac{\#h}{\#j}$, #j is the number of users who rated item j, #h is the number of users who rated the item *j* with the rating value *h*. Although there is no overlap in the rating vectors I and J, the BC measure still captures the overall similarity between two items. The local correlation $loc_{cor}(r_{ui}, r_{vj})$, between two users is given by:

$$loc_{cor}(r_{ui}, r_{vj}) = \frac{(r_{ui}-\bar{r}_u)(r_{vj}-\bar{r}_v)}{\sigma_u\sigma_v} \quad (12)$$

It provides local user information i.e. the similarity of users in rating behaviour even if there are no co-rated items between user *u* and *v*. The local similarity can provide positive as well as negative correlations. This technique is useful especially when the user-item matrix is quite sparse. Although this measure can utilize all ratings to measure similarity in a sparse dataset, there may be reduced accuracy due to the absence of overlapping items. To provide more importance to the number of common items, Jaccard Similarity between the user *u* and *v* is added with (10). The modified equation is as follows:

$$S(u,v) = Jacc(u,v) + \sum_{i\in I_u}\sum_{j\in I_v} BC(i,j)loc(r_{ui}, r_{vj}) \quad (13)$$

2.1.6 Jaccard Index

The Jaccard index which is also known as the Jaccard similarity coefficient is a statistic that is used for comparing the similarity and diversity of sample sets. Given two users *u* and *v*, each with *n* binary attributes, the Jaccard coefficient measures the degree of overlap between the two sets by dividing the numbers of items observed by both users (intersection) and the number of different items from both sets of rated items (union).

$$J(u,v) = \frac{|u \cap v|}{|u \cup v|} \quad (14)$$

A complementary measure of similarity is the Jaccard distance, which measures the dissimilarity between sample sets is given by either subtracting the Jaccard coefficient from 1, or, equivalently, by dividing the difference of the sizes of the union and the intersection of two sets by the size of the union: $d(u,v) = 1 - J(u,v)$

$$= \frac{|u \cup v| - |u \cap v|}{|u \cup v|} \quad (15)$$

This similarity metric only considers the number of items that have been rated in common and ignores rating values. Studies have shown that this metric is advantageous when dealing with an extremely asymmetric or sparse datasets [28]. Jaccard coefficient is a good measure when the datasets are binary, however, if the values of the user interest in products are detailed ratings reflecting the degree of user interest, then the Jaccard distance fails to capture important information [5].

2.2 Prediction

The rating prediction of the target user for potential item of interest is then based on partial information of the active user (ratings) and a subset of weights as calculated in the above given equations.

For the correlation based similarity approach, vector similarity and the union based approach, prediction can be defined as:

$$p_{uj} = \bar{r}_u + \alpha\sum_{v\in U_j} w(u\,v)\left(r_{vj} - \bar{r}_v\right) \quad (16)$$

where p_{uj} is the predicted rate for user *u* for item *i*, U_j denotes the set of k-nearest neighbours most similar to user *u* and have rated item *j*, $w(u\,v)$ is the computed similarity between user *u* and user *v* as given in (1) and $\alpha = \frac{1}{\sum_{v\in U_j}|w(uv)|}$ is a normalization factor such that the absolute values of the weights sum to unity.

For the item-based similarity approach, prediction is made by calculating the simple weighted average of the item similarities and is given by:

$$p_{uj} = \frac{\sum_{i\in I_u} r_{ui}w_{ij}}{\sum_{i\in I_u}|w_{ij}|} \quad (17)$$

where I_u is the set of items user u has rated , w_{ij} is the weight between items i and j as given in (6), r_{ui} is the rating for user u on item i [11, 21].

For the slope one predictor method, the prediction process involves taking the average of all deviations. Additionally, to take into account the number of ratings observed, the weighted slope one prediction is defined as follows:

$$p_{uj} = \frac{\sum_i I_u \left(r_{ui} + dev_{ij}\right)|U_{ij}|}{\sum_i |U_{ij}|} \quad (18)$$

where $p_{u,j}$ is the prediction made for item j for user u, U_{ij} is the set of users who have rated both items i and j, dev_{ij} is given in (7). It was shown that the Slope one predictor overcomes some of the limitations of the item-based and user-based similarity method. It aims to provide valid recommendations even for users who have few ratings, improved time efficiency and scalability [6, 24].

2.2.1 Regression and Classification

In equations (17), (18) and (19), the computed similarity value is combined with the user ratings to assign appropriate weights to the contribution of every neighbour since users can have different levels of similarity to a target user. These equations are based on regression. Regression based method is suitable for continuous values of user feedback.

In our paper, any type of user rating is mapped to three values 0, 0.5, and 1. In such cases, rating prediction is done by classification by having the nearest neighbours of the target user vote on this value. The vote (v_{ir}) can be obtained as the sum of the similarity weights of neighbours that have given this rating to item i..

$$v_{ir} = \sum_{v \in U_i} \delta(r_{vi} = r)\, w_{uv} \quad (19)$$

where $\delta(r_{vi} = r)$ is 1 if $r_{vi} = r$ and 0 otherwise

2.3 Generating Top-N Recommendations

In the earlier recommender systems like Tapestry and GroupLens, the predicted ratings for potential items of interest were displayed. Users used the displayed predicted ratings to decide which items to choose. However, very quickly researchers recognized the importance of considering recommendation lists where items of interest are displayed as a list as opposed to only displaying predicted ratings [1, 29]. Breese et al [12] introduced a ranked scoring metric that recognizes the value of a recommendation as a function of its position in the list and the utility of the item. It is based on the evidence that the likelihood of selecting a recommended item decays exponentially as the item is lower on the list.

2.4 Evaluation Metrics

To assess the quality of the predicted recommendations, the system has to be evaluated. The type of evaluation metrics used depends on the type of CF application [11]. According to [29], evaluation metrics can be broadly classified into three categories: *predictive accuracy metrics, classification accuracy metrics, rank accuracy metrics.* Predictive accuracy metrics measure how close the recommender system's ratings are to the true user ratings. The most commonly used evaluation metric in this category is Mean Absolute Error (MAE). This metric computes the average of the absolute difference between the predictions and true user ratings. It is given by: $\frac{1}{|I_u|} \sum_{i \in I_u} r_{ui}$

$$MAE = \frac{\sum_{I_u} |p_{ui} - r_{ui}|}{n} \quad (20)$$

where n is the total number of ratings over all users, p_{ui} is the predicted rating for user u on item i, r_{ui} is the actual rating of user u on item i, I_u is the set of items rated by user u. The lower the value of MAE, the better the prediction. A variation of MAE that is widely used Root Mean Squared Error (RMSE). Here, the difference between the predicted rate and true rate is squared before summing it in order to put more emphasis on large errors.

$$RMSE = \sqrt{\frac{1}{n} \sum_{I_u} (p_{ui} - r_{ui})^2} \quad (21)$$

where n is the total number of ratings over all users, p_{ui} is the predicted rating for user u on item i, r_{ui} is the actual rating of user u on item i., I_u is the set of items rated by user u.

Classification accuracy metrics do not measure the ability of an algorithm to accurately predict ratings rather measure the frequency with which a recommender system makes a correct or incorrect decision about whether an item is good. It is used when the user responses are binary. Rank accuracy metrics measure the ability of a recommendation algorithm to produce a recommended ordering of items that matches how the user would have ordered the same items. Unlike classification metrics, ranking metrics are more appropriate to evaluate algorithms that will be used to present ranked recommendation lists to the user, in domains where the user's preferences in recommendations are non-binary [29].

3. ANALYSIS AND DISCUSSION

Neighborhood based collaborative filtering has been a widely used recommendation technique in various e-commerce applications. The different techniques in the neighbourhood based approaches compute similarity between users or items by representing user profiles as a vector of ratings given to individual products. They then use the weighted average of this vector of ratings to make predictions and item recommendations. The user-user based collaborative filtering algorithm based on Pearson Correlation Coefficient is recognized widely as providing high quality recommendations. However, they face challenges of data sparsity and scalability. As an alternative the item based collaborative filtering recommender system was developed that builds correlations between pairs of items instead of users. This approach is capable of providing a better scalability since once similarity is computed between two items, it need not be repeated for a period of time. Item-based similarity approach is especially useful in e-commerce stores that have a large number of items that are more stagnant than the users. However, the item-based approach has the disadvantage that the list of recommended items mostly contained those items that the user appreciates often. Businesses did not want to waste recommendations on a product customers would likely purchase anyway. Hence for a more serendipitous recommendations, the user based collaborative algorithm is preferred. Both user based algorithms and item based algorithm cannot make recommendations to a user who is new to

the system and has rated only few items. In such cases, the slope one predictor algorithm is quite useful in that it makes recommendations based on the general popularity of the item therefore a user with few ratings should also receive valid recommendations.

A collective drawback of all the aforementioned techniques is that they can make recommendations only if there are some commonly rated items between any two users. In a product database where there are millions of items, rating data as such can be very sparse. Finding items that have been commonly rated by two users is therefore very scarce. As a result, calculating a prediction and recommendation based on such sparse data will not be very reliable and requires further improvement. Secondly, the ratings tell us nothing about a user's comprehensive interest. If the ratings were a reflection of interest scores given to product categories instead of individual products, then the user's overall interest can be understood better, which will enable the generation of more meaningful similarity computations between individuals or items.

4. OVERVIEW OF THE PROPOSED TECHNIQUE

Following the study of similarity measures and the observations made, we propose the hypothesis that by categorizing the products in a hierarchy, some of the issues in the existing system can be resolved. Firstly, given a set of users and products, the products are divided into categories and sub-categories represented by a tree. Then based on the rated items, a weight is assigned to all unrated items (the leaf nodes). In this way for each user u, we will have a set of weights for all product items w_1, w_2, ..w_m, where 'm' is the total number of items. A rated product will make a contribution to all non-rated items. For example, if 'i' is an item rated by a user, its contribution to an unrated item 'j' by the same user will be 1/d, where d is the distance between item 'i' and item 'j'. In a similar fashion, item 'j' will receive contributions from all rated items. These contributions should be summed up and normalized to represent the user's interest in the unrated item 'j'. We will call this 'potential interest score' of user u. Secondly, similarity between pairs of items is computed. In order to measure similarity between items even in the absence of a user who has rated both items, the BCF measure is used to compute similarity between two items. The BCF measure combines the general rating information of two items with its local rating correlation as given in equation 11 and 12 respectively. We make a slight modification to equation 12.

$$loc_{cor}(r_{iu}, r_{jv}) = \frac{(r_{iu} - \bar{r}_i)(r_{jv} - \bar{r}_j)}{\sigma_i \sigma_j} \quad (22)$$

Similarity between item I and item j in BCF measure is as follows:

$$S(i,j) = \sum_{i \in I_u} \sum_{j \in I_v} BC(i,j) loc(r_{iu}, r_{jv}) \quad (23)$$

The BC coefficient between a pair of rated items measures the similarity of two items in a global perspective. The importance of the local similarity is increased or decreased depending on the value of the BC coefficient, which provides maximum importance to local similarity if ratings are made on the same item, as similarity on the same item is 1 (BC(i, i) = 1). On the other hand if the BC coefficient between two items is 0, then it gives no importance to the local similarity. In order to give more value to the number of common items, the Jaccard similarity measure is added with equation 24. The 'potential interest score' of a user 'u' in item j is combined with equation 24, where the BCF measure gives the similarity between items I and j by way of their ratings and the potential interest score gives value of user interest in the item j. Prediction value for item j can be calculated as:

$$p_{uj} = \frac{\sum_{j \in items\ similar\ to\ i} r_{ui} w_{ij}}{\sum_{j \in items\ similar\ to\ i} |w_{ij}|} \quad (24)$$

where w_{ij} is a product of equation 24 and the potential user interest.

It is our hope that this technique will a) alleviate the sparsity problem to an extent b) capture the comprehensive interest of a user and thereby produce more reliable recommendations.

5. CONCLUSION

Recommender systems are an important tool in e-commerce that assist the customers in buying products. This paper discusses the different similarity measures used in neighbourhood based collaborative filtering recommender system. It also highlights other factors that play a part in the overall efficiency of the recommendations generated. We have proposed that categorizing items in a hierarchy would enable to predict ratings for all unrated items thereby making the user-item matrix denser. Moreover this way the user interest can be captured more fully. We believe that this approach will provide users with more informed recommendations.

REFERENCES

1. Joseph A. Konstan, J.R., Recommender Systems: from Algorithms to User Experience. User Modeling and User Adapted Interaction, 2012. 22: p. 101 - 123.
2. Cho, Y.H. and J.K. Kim, Application of Web Usage Mining and Product Taxonomy to Colaborative Recommendations in E-commerce. Expert Systems with Applications, 2004: p. 233-246.
3. Baum, D. and M. Spann, The Interplay between Online Consumer Reviews and Recommender Systems: An Experimental Analysis. International journal of electronic commerce, 2014. 19(1): p. 129-162.
4. Desrosiers, C. and G. Karypis, A Comprehensive Survey of Neighbourhood-based Recommendation Methods, in Recommender Systems Handbook. 2011, Springer.
5. Lescovec, J., A. Rajaraman, and J.D. Ullman, Mining of Massive Datasets. 2014.
6. Linyuan Lu, M.M., Chi Ho Yeung, Yi-Cheng Zhang, Zi-Ke Zhang, Tao Zhou, Recommender Systems. Physics Reports, 2012. 519(1): p. 1 - 49.
7. Burke, R., Knowledge-Based Recommender Systems, in Encyclopedia of Library and Information Systems. 2000, Marcel Dekker.
8. Park, D.H., et al., A Literature Review and Classification of Recommender Systems Research. Expert Systems with Applications, 2012: p. 10059-10072.
9. Koren, Y. and R. Bell, Advances in Collaborative Filtering, in Recommender Systems Handbook. 2011, Springer US. p. 145-185.

10. Shardanand, U. and P. Maes. Social information filtering: lgorithms for automating "word of mouth". in Conference on Human Factors in Computing Systems. 1995.
11. Su, X. and J.M. Khoshgoftaar, A Survey of Collaborative Filtering Techniques. Advances in Artificial Intelligence, 2009.
12. Breese, J.S., D. Heckerman, and C. Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. in Conference of Uncertainty in Artificial Intelligence. 2013.
13. Kim, H.K., J.K. Kim, and Y.U. Ryu, Personalized Recommendation over a Customer Network for Ubiquitous Shopping. IEEE Transactions on Services Computing, 2009. 2(2): p. 140-152.
14. Sollenborn, M. and P. Funk, Category-based Filtering and User Stereotype Cases to Reduce the Atency Problem in Recommender Systems. in Proceedings of the Sixth European Conference on Case-based Reasoning. 2002.
15. Melville, P., R.J. Mooney, and R. Nagarajan, Content-boosted Collaborative Filtering for Improved Recommendations, in Eighteenth National Conference on Artificial Intelligence. 2002. p. 187 - 192.
16. Resnick, P., et al., GroupLens: An Open Architecture for Collaborative Filtring of Netnews. Computer Supported Co-operative Work, 1994: p. 175-186.
17. Adomavicus, G. and A. Tuzhilin, Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extension. IEEE Transactions on Knowledge and Data Engineering, 2005. 17: p. 734-749.
18. Hu, Y., Y. Koren, and C. Volinsky, Collaborative Filtering for Implicit Feedback Datasets, in Eighth IEEE International Conference on Data Mining. 2008. p. 263 - 272.
19. Lee, T.Q., Y. Park, and Y.-T. Park, A Time-Based Approach to Effective Recommender Systems using Implicit Feedback. Expert Systems with Applications, 2008. 34: p. 3055-3062.
20. Goldberg, D., et al., Using collaborative filtering to weave an information tapestry. Communications of the ACM - Special Issue in Information Filtering, 1992: p. 61-70.
21. Sarwar, B., et al. Item-Based Collaborative Filtering Recommender Algorithms. in Conference on World Wide Web. 2001. Hong Kong.
22. Symeonidis, P., et al., Collaborative Filtering: Fallacies and Insights in Measuring Similarity, in Tenth European Conference on Principles and the Practice of Knowledge Discovery in Databases Workshop on Web Mining. 2006. p. 56-67.
23. Ekstrand, M. Similarity function for user-user collaborative filtering. 2013.
24. Lemire, D. and A. Maclachlan. Slope One Predictors for Online Rating-based Collaborative Flitering. in Proceedings of SIAM Data Mining. 2005.
25. Zacharski, R., A Programmer's Guide to Data Mining. 2012.
26. Deshpande, M. and G. Karypis, Item-Based Top-N Recommendation Algorithms. ACM Transactions of Information Systems, 2004. 22: p. 143-177.
27. Patra, B.K., et al., A New Similarity Measure using Bhattachryya Coefficient for Collaborative Filtering in Sparse Data. Knowledge-Based Systems, 2015. 83: p. 163-177.
28. Verbert, K., et al. Dataset-driven Research for Improving Recommender Systems for Learning. in First International Conference on Learning Analytics and Kowledge. 2011. ACM.
29. Herlocker, J.L., et al., Evaluating collaborative filtering recommender systems. ACM Transactions of Information Systems, 2004. 22: p. 5-53.
30. Zieglar, C.-N., G. Lausen, and J.A. Konstan, On Exploiting Classification Taxonomies in Recommender Systems. AI Communications, 2008. 21: p. 97-125.